Anonymity in Wireless Broadcast Networks

Matt Blaze[1], John Ioannidis[2], Angelos D. Keromytis[3], Tal Malkin[3], and Avi Rubin[4]
(Corresponding author: Angelos D. Keromytis)
Computer and Information Science Department, University of Pennsylvania (Email: blaze@cis.upenn.edu)[1]
Consultant (Email: ji@tla.org)[2]
Department of Computer Science, Columbia University[3]
MC 0401, 1214 Amsterdam Avenue, New York, NY 10027, USA (Email: angelos@cs.columbia.edu)
Computer Science Department, Johns Hopkins University (Email: rubin@cs.jhu.edu)[4]
(Received Aug. 20, 2007; revised Dec. 5, 2007; and accepted Jan. 21, 2008)

Abstract

Systems that provide network traffic anonymity typically focus on wide-area network topologies, and exploit the infeasibility of eavesdropping on all links to prevent attackers from determining communication peers. This approach is inappropriate for high-security wireless local-area networks, since it does not obscure the traffic volume, allowing attackers to identify critical nodes (*e.g.,* a military HQ) and, given the ability of an attacker to obtain a global view of all communications, the relative ease of identifying the source and destination of traffic flows. These weaknesses derive from the fact that, whereas in wide-area networks the sender, the receiver and the adversary are on different physical links, in wireless networks they may share a single broadcast link. Moreover, the adversary can easily find the physical location of the transmitter and thereby identify the entity sending the traffic, not just its network identity. We introduce *Wireless Anonymous Routing (*WAR*)*, an approach to achieve anonymity in a broadcast network. We describe a formal threat model for WAR and compare it to the traditional anonymity approaches. We show that these are inadequate when applied to the broadcast model, and describe new protocols that preserve security with better performance, adequately addressing the requirements of security-critical environments. We provide analytical and some preliminary experimental evidence that our protocols achieve anonymity at a reasonable cost.

Keywords: Anonymity, MANET, onion routing, source routing, wireless

1 Introduction

Anonymity and resistance to traffic analysis is an interesting and difficult problem in computer networking. In most modern networks, including IP-based ones, communication peers inherently identify the sources and destinations of traffic to routers, gateways, and ancillary servers. In packet-switched networks, the network-level addresses are visible to anyone with access to any link over which the traffic flows. An especially difficult aspect of this problem involves hiding various aspects of the identity of peers from each other.

A number of schemes for anonymity have been developed in recent years; most are variants on "mix networks" [13], in which traffic is routed among participants in the network in an effort to hide the true source and/or destination of the messages. These schemes vary in details concerning such issues as whether and between whom cryptography is used, how membership is managed, at which layer of the protocol stack they operate, whom the identity is hidden from, and so on. What most of these schemes have in common is the assumption (and exploitation of the fact) that they operate in a wide-area network with multiple links, where it is infeasible for the adversary to monitor all links and to obtain global information about network traffic.

This assumption excludes an important class of networks, one in which many interesting requirements for anonymity may be found: broadcast (typically wireless) communication systems. Here, not only can the adversary usually listen to all traffic, but he can also identify the physical location of the sender by using radio-frequency direction-finding techniques. While the sender may be almost impossible to conceal, the receiver, being generally "passive" need not reveal his identity, or even his presence, to anyone in order to receive a message. In such environments, it is also relatively straightforward to identify communication hubs, such as a command-and-control post; these may be singled out for directed attacks.

These differences break (or render terribly inefficient) the standard anonymity protocols designed for large networks. At the same time, the broadcast nature of such networks tends to exacerbate privacy and traffic analysis issues, which tend to be either of no concern or of extreme importance to the users of the network, especially in applications such as military communications, some types of sensor networks, wireless mesh networks, *etc.*

In this paper, we advance the notion that wireless networks (*e.g.,* sensor or *ad hoc* networks) are an interesting and not well explored research area for anonymity and identity protection. In particular, we introduce a security model for privacy in such networks, a suite of protocols that provide anonymity under the model, and analytical and experimental results that suggest that these protocols are useful in specific scenarios where the tradeoff between anonymity and performance are—within reason—weighed toward the former, *e.g.,* a battlefield wireless communication network. We also suggest open problems and directions for future work.

The remainder of this paper is organized as follows: Section 2 presents our model, including the communication infrastructure and the security model. Section 3 presents our protocol variants and their security properties. We discuss our prototype implementation in Section 4 and show some preliminary performance results in Section 5. Section 6 gives an overview of related work.

2 The WAR Model

2.1 Communications Infrastructure and Environment

We consider a communication network where stations can communicate only through a public broadcast (usually wireless) channel. Every station listens for transmissions in "promiscuous mode," meaning they can receive the transmissions of every other station. Stations can exhibit a high degree of mobility, and can join and leave the network at any time. Some stations may have a gateway to another network, such as the Internet.

For simplicity, we assume a wireless broadcast medium in which only one station at a time can transmit a message, which is potentially received by all other stations within communications range; receipt of a particular transmission by any given station is not guaranteed by the medium itself, however. As a consequence of transmitting, the station may reveal identifying information (such as its physical position or unique transmitter characteristics, *e.g.,* the exact oscillation frequency of its crystals) about itself. Receiving a message, on the other hand, is entirely passive, and does not reveal any information about the receivers. Every station can act as both a transmitter and a receiver, but is identified to others only when it transmits. Note that "Tempest"-type attacks against receivers can invalidate this assumption by revealing information about the internal state of processors, but we do not consider such attacks here. In other words, we assume abstract characteristics similar to those of most conventional general-purpose radio-based schemes, including commercially available wireless networks such as 802.11 systems operating in "ad hoc" mode. The network abstraction we are providing is that of an unreliable packet local network, on top of which we could run IP. The service characteristics may not be suitable for all higher-layer protocols, as the performance results we get for TCP show in Section 4. Part of our future work includes extending our model to take into consideration real-world wireless phenomena [1, 19].

We refer to packets transmitted by the stations as *radiograms*, to distinguish them from IP packets or higher level message abstractions. Each radiogram is sent by a single *sender*, and can be intercepted by anyone in range. Every radiogram consists of two parts: a sender ID, which contains the public key of the transmitting station, and an encrypted payload. This requirement is made without loss of generality, because our threat model will assume that the adversary can always identify the sender of a radiogram (as this is hard to conceal in a wireless network). Thus, adding the ID does not help the adversary, but can be used by the "good" stations to identify the sender, and maintain a list of current group members. We assume that the payload is encrypted using a symmetric or asymmetric encryption scheme that should satisfy the standard notion of semantic security [26], and that it contains enough redundancy to know unambiguously whether an attempt to decrypt it with a certain key has succeeded. Therefore, we can associate with each radiogram a set of *receivers*, which are those stations that can successfully decrypt the payload[1]. Finally, we assume for simplicity that there is a fixed radiogram size, with short payloads padded out to this size. The model can easily be extended to support a fixed number of different radiogram sizes.

Consider a station s_i that wants to send a message m (such as an IP packet) to station s_j. We refer to s_i as the *originator* of m and s_j as the *target* of m. A Wireless Anonymous Routing (WAR) protocol specifies how s_i should originate such communication, decide which other stations will be involved, and the series of radiograms that need to be transmitted by these stations so that m will eventually reach s_j. For analysis purposes, each radiogram sent in a WAR protocol will be associated with one corresponding high-level message, or considered a *cover* radiogram if it is not associated with any message.[2] The sender of a radiogram does not necessarily know (nor should it) whether the radiogram is cover traffic or not, what message it is associated with, or who the receivers of the radiogram are. We refer to all high-level messages that are being delivered among originator/target pairs as the *core traffic* in the system.

Our only formal correctness requirement will be that if s_i originates the delivery of m to s_j, then s_j will eventually receive m (so long as neither of them has left the group of active stations and the network is not saturated). Clearly, to make the protocol usable, mere delivery is not enough, and round trip time, as well as other performance parameters, should be reasonable. The security requirements of a WAR protocol are discussed below.

[1] In our protocols, each radiogram will have a single successful receiver.

[2] Practical optimizations such as sending several short messages in a single radiogram are beyond the scope of this paper.

2.2 Adversary Model

We consider two types of adversaries: a *listening adversary* and a *Byzantine adversary*, each of which can be either *non-adaptive*, *adaptive*, or *pro-active* (following the standard cryptographic notions).

A listening adversary can monitor *all* traffic, and identify the sender of each radiogram. In addition, the adversary is able to monitor the internal state of a certain set of (compromised) stations (including secret keys, randomness, computation of radiogram to be sent, *etc.*). This adversary is honest-but-curious, in that he follows the protocol in each station, but tries to deduce information in order to compromise anonymity. The set of compromised stations may be restricted to be in some access structure (a subset of the powerset of the set of all stations), such as "all subsets of at most five stations." To capture a simpler adversary who is capable of listening only to transmissions, and does not have access to any other station's internal state, we can take the set of internally compromised stations to be the empty set.

A Byzantine adversary is stronger. Like the listening adversary, it can monitor all traffic and the internal state of some set of stations in the access structure. In addition, it can also maliciously control up to t stations, for some parameter t. The adversary can make controlled stations behave arbitrarily, including injecting new radiograms, dropping radiograms, or changing radiograms that were supposed to be transmitted.

We can make a further (by now standard) distinction between a non-adaptive, adaptive, or pro-active adversary, depending on when the corrupted stations are chosen. A non-adaptive adversary chooses the set of stations to be corrupted (and the t stations to be controlled in the Byzantine case) at the beginning of his attack. An adaptive adversary may choose which stations to corrupt as his attack progresses, based on the information gathered so far (but without violating the access structure). A pro-active adversary can adaptively choose which stations to corrupt, but can also choose (synchronously) to shift its corruption from a certain station to another, granting the previously corrupted station healed. Here, the only restriction is that the adversary does not violate the access structure in any particular time period. More details on pro-active security (in a different context) may be found in [10, 31, 45].

2.3 Security Goals

Our goal is to maintain anonymity of both originator and target of a given message. Clearly, if an adversary corrupts the originator, he can see the originated message m and its target.[3] It is also clear that if the adversary corrupts the target station, he can figure this out upon delivery of the message. Anonymity should be maintained for all other cases (including anonymity of the originator when the target is corrupted). These main anonymity goals are formalized below for a listening, non-adaptive adversary. A discussion of stronger adversaries and additional goals follows.

Let S be the set of all stations, 2^S its powerset, $\mathcal{C} \subseteq 2^S$ an access structure, and $C \in \mathcal{C}$ the set of compromised stations by a listening adversary. Denote its view, after an execution of a protocol in which an originator s_i delivers a message to a target s_j, by the random variable $V_C^k(i,j)$, where k is the security parameter. This view contains all radiograms sent in the network, and the internal state of all nodes in C.

Originator and Target Anonymity to Observer: We require that for any message delivered, as long as the originator and target are not corrupted, the adversary's view reveals no information about their identity. That is, for all $i_1, j_1, i_2, j_2 \notin C$, $\{V_C^k(i_1,j_1)\}$ and $\{V_C^k(i_2,j_2)\}$ are computationally indistinguishable.

Complete Anonymity. A stronger requirement is that as long as the originator is not corrupted (even if the target is), the adversary's view reveals no information about the identity of the originator. If the target is not corrupted then no information about its identity should be revealed either. Note that if the target (final destination of an anonymous message) is corrupted, the message can be decrypted, which might contain the identity of the originator, *e.g.,* as part of network protocol headers (we do not restrict the content of messages!). Even in that case, however, the adversary will not know which radio device corresponds to a given originator. Thus, in formalizing this goal we use a randomized message. We denote by $RV_C^k(i,j)$ the randomized view of the adversary after an execution of a protocol in which an originator s_i delivers a uniformly chosen message to target s_j. We require that originator and target anonymity to observer is maintained, and that in addition, for all $i_1, i_2 \notin C$ and for all $j \in C$, we have that $\{RV_C^k(i_1,j)\}$ and $\{RV_C^k(i_2,j)\}$ are indistinguishable.

An additional goal which may be desired, is that of **hiding the existence of communication**. That is, an adversary cannot even determine whether any communication is taking place (unless he has corrupted an originator or target of such communication). Instead of formalizing this goal directly, we formalize an even stronger goal which implies it, is achieved by our main protocol, and has other advantages.

Anonymity Per Radiogram. For each radiogram r that is sent during protocol execution, let s_i, s_j be correspondingly originator and target (where $s_j = \perp$ for a cover radiogram). We require that, as long as $s_i, s_j \notin C$, the adversary's view gives him no information about s_j. In particular, the adversary cannot distinguish the cases where r is a cover radiogram or a radiogram that can be decrypted by a certain s_j.

[3] The originator must know the target to which he sends a message. But, this target may be specified in terms of a public key that is a pseudonym of an unknown party, such as a public key that was previously sent within an anonymous message.

We note that anonymity per radiogram should not be viewed as a required goal, since the real concerns are with the core traffic (high-level messages). However, this is a useful tool since, if achieved, it can be proved to imply originator and target anonymity to observer, as well as hiding the existence of messages.

Security against Stronger Adversaries. We defined our anonymity goals for a listening, non-adaptive adversaries, and our analysis will concentrate on this setting as well. The formal definitions for adaptive or pro-active adversaries, and for Byzantine adversaries are quite complex, and are not included here (some of the issues are similar to those in defining secure computation against active adversaries). For example, if the adversary is adaptive, it is not enough to ask that originator and target remain anonymous as long as the adversary has not corrupted them, since the protocol might allow an adversary to adaptively corrupt parties so that the originator and target are always corrupted with high probability. Such an attack should be prohibited by the definition. Of course, for certain protocols it may be possible to show that an adaptive or pro-active adversary cannot gain more information than a non-adaptive adversary.

We note that a general transformation from a protocol secure against a listening adversary to one secure against a Byzantine adversary, can proceed by requiring each radiogram's payload to be authenticated with respect to the public key in its first ID part (and discarding radiograms that fail to authenticate). The anonymity features will be preserved, although reliable delivery may no longer be guaranteed (*i.e.,* the system may become susceptible to denial-of-service attacks). Reliability can be addressed by other means, and is not the focus of the current work.

2.4 Discussion

Our anonymity goals are strong, and should be achieved regardless of the core traffic characteristics. To achieve our strongest goals (including hiding the existence of communication), the pattern of radiograms to be sent by a node should be independent of the volume, content, or any other aspect of the core traffic. That is, the protocols should specify the size and frequency of radiograms to be sent at each node, *e.g.,* by sending at a constant rate (which may depend on the overall traffic volume and group size, but not the core traffic). To an observer, the distribution of these radiograms will look the same throughout the protocol.

We note that weaker goals of anonymity were sometimes posed by previous works, such as mix-based mechanisms (see Section 6). For example, anonymity may be achieved based on statistical properties of the core traffic (requiring high volume). Or it may be allowed to leak the existence or even content of messages, as long as there is no linkability between a message originated by a source and a message delivered to a destination. We do not elaborate on these weaker goals as our work provides stronger security guarantees.

Finally, we should mention that if an adversary can gain arbitrarily many identities (public keys), then it is clearly easy for him to control a large number (or fraction) of nodes. This problem is outside of our model, and should be prevented at the public-key infrastructure level. However, we note that if a protocol achieves anonymity against an arbitrary access structure $\mathcal{C}$ (without a size limitation), then this attack does not help the adversary. Indeed, this is the case with our main protocol, which achieves source and destination anonymity to observer for arbitrary $\mathcal{C}$.

3 Protocols and Analysis

3.1 Basic Protocols

Here, we describe some basic WAR protocols, which demonstrate general solutions and ideas that will be used by our main construction in the Section 3.2. In all the protocols, a station joins by broadcasting its public key, and then listens for a while to hear public keys of other (existing and new) members.

3.1.1 From Strong Anonymous Routing to WAR

Previous practical approaches and systems to anonymous routing in networks require varying anonymity properties, under different adversarial models. Almost all these adversarial models consider an adversary that has access only to part of the network links, and cannot obtain global information about the network traffic. However, some works (*cf.* [2]) require anonymity against an adversary that can monitor *all* traffic, like our listening adversary. We refer to such protocols as strong anonymous routing protocols.

It easy to see that such protocols can be transformed to our wireless setting, where instead of sending the radiogram r on a link from station i to j, we have station i broadcast the payload $(j, E(PK_j, r))$, where $E(PK_j, r)$ is the (asymmetric) encryption of r under the public key of station j. Such a transformation preserves the anonymity guarantees, although not necessarily reliability (as the network is dynamic and stations leave as they please). The transformation also maintains the performance of the original schemes, with respect to a clique network topology. This transformation provides several candidate WAR protocols, such as those adapted from variants of Beimel and Dolev [2].

However, these protocols have weaknesses in terms of performance and security; since such protocols were designed for a fixed point-to-point network, they cannot take advantage of the broadcast channel, and thus impose higher communication overheads. Intuitively, these protocols obtain anonymity *despite* the broadcast channel, but cannot use it for improved performance. Such protocols may also exhibit particularly bad performance in our clique setting, *e.g.,* if performance depends on the

number of edges in the network. In terms of security, these protocols achieve anonymity against a listening-only non-adaptive adversary. They do not achieve complete source anonymity, and their corruption threshold under which anonymity against adaptive adversary is achieved is worse than our constructions below.

3.1.2 From Anonymous Routing to WAR

Starting from any anonymous routing protocol for a point-to-point network (such as a mix or onion routing), we can transform it to a WAR protocol by using generic mechanisms to create virtual point-to-point links out of the broadcast channel, and running the given protocol on top of them. Some more details follow.

As a first step, we can use the broadcast channel to implement secure channels for each pair of nodes[4]. This simply reduces to key exchange among each pair (*cf.* [11, 12]), where the key is then used for symmetric encryption of radiograms between the pair of nodes. For example, we can use a Diffie-Hellman-based key exchange, ending up with a secret key of the form g^{xy} for each of the n^2 pairs of current members. Alternatively, instead of establishing n^2 secure channels, public key encryption can be used, where to send a message to s_j, the encryption of the message under s_j's public key is broadcast. Here it is essential to use an encryption scheme providing *key-privacy* [4], namely where a ciphertext does not reveal under which key the message was encrypted. While standard RSA does not satisfy this, several variations of RSA, as well as other encryption schemes, are known to satisfy key-privacy [4]. By directly broadcasting the message, this already guarantees originator and target anonymity to an observer (though the target will know who the originator was). For complete anonymity, the following step is taken.

As a second step, we use the established (virtual) secure channels to implement the point-to-point protocol. Namely, instead of sending the radiogram r on a link from station i to j, we have station i broadcast the payload $E(sk_{i,j}, r)$, where $E(sk_{i,j}, r)$ is the (symmetric) encryption of r under the secret key of the pair (i, j); or, for the public-key alternative, station i broadcasts the payload $E(PK_j, r)$, which is the (asymmetric) encryption of r under the public key of station j. For each broadcast radiogram from node i, every node j tries to decrypt the payload using its key. If the decryption fails, the radiogram is discarded (as it was intended for a different receiver). If the decryption succeeds, the node continues as specified by the underlying protocol.

For example, we briefly describe the scheme using symmetric keys established through key-exchange, and the onion routing [49] protocol. A node that wants to send a message to some group member, chooses at random a set of $M-1$ current group members, where M is a security parameter. It then creates an "onion" of M encrypted encapsulated payloads. The innermost layer is encrypted with the public key of the intended destination. Each successive layer consists of an (asymmetric) encryption of the previous layers' payload together with the identity of the corresponding group member, under the public key of the next group member. Finally, the onion is encrypted using the symmetric key of the intended last group member in the chain, and broadcast.

Upon broadcast of a radiogram by station i, all stations try to decrypt it (using their symmetric keys with i), but only one station (the receiver) succeeds. The receiver further decrypts using its asymmetric private key, to find a payload r consisting of an index j and an encapsulated payload r', requiring the receiver to encrypt r' using its secret key with node j, and broadcast the resulting payload. This continues until the innermost layer is reached (in which case the payload is decrypted to the actual message). Note that this protocol involves only a symmetric decryption operation for each station, per broadcast radiogram, which is quite fast (if we were to use public key encryption on the last layer as well, each node would have to apply heavy asymmetric decryption on each broadcast radiogram, though the legitimate receiver would not have to perform further symmetric encryption/decryption).

Anonymity properties are inherited from the underlying protocol. In particular, using onion-like mechanisms (similar to those used in Mixmaster or Mixminion) with an onion of depth M, it can be shown that complete anonymity is guaranteed as long as there are two consecutive uncorrupted nodes in the used onion, and in particular, as long as the adversary has corrupted at most $M/2$ onion nodes in total. This is because each node knows the identity of its onion-neighbors (*i.e.,* the sender of the previous radiogram, and the receiver of the next radiogram). This implies that for adaptive adversaries, anonymity is maintained as long as fewer than $M/2$ are corrupted (thus, for a given maximum number t of corrupted nodes, we can choose $M > 2t$ to be the length of the onion). For a non-adaptive adversary we can choose a smaller M, since even if the adversary corrupts $t > M/2$ nodes, the probability that these will contain $M/2$ of the nodes in a randomly chosen onion is still rather small.[5]

3.1.3 WAR Using Public-Key Cryptography

We now describe a protocol that achieves the best security guarantees of all our protocols. However, its performance is prohibitive for real applications, as we will see in Section 5. This will be fixed by our main construction in Section 3.2, which uses the protocol described in this section for symmetric-key distribution (making its performance characteristics less critical).

Consider again an anonymous point-to-point protocol, such as onion routing. We propose a modification to the protocol above, in which senders of radiograms do not generally know who the receivers are (thus allowing for better anonymity properties). Specifically, the protocol

[4] An authenticated broadcast channel is required, and be achieved using the public-key infrastructure of joining members.

[5] The exact expression is $[\Sigma_{i=m/2}^{t}\binom{t}{i}\binom{N-t}{M-i}]/\binom{N}{M}$.

proceeds similarly to the one above, except that the onion does *not* include the identity of the receiver for each layer, thus departing from the traditional point-to-point onion (we call this an undirected onion). We use the public-key version here (all encryption and decryptions are asymmetric), and combine this heavy weight protocol with a much more efficient one in the next section. We provide some more details about the protocol below, as it will be used as a sub-protocol for our main construction.

There are three kinds of payloads in this protocol:

- *Cover.* Cover payloads are just random bit-strings, and are sent any time a node wants to send a radiogram but has no outgoing traffic in its queue. The goal of cover traffic (a form of *traffic padding*, similar to that used by MIXes [13], which is further discussed in Section 6) is to conceal the transmission of real traffic (whether originating at the node or relayed through it), and thus defeat timing or other inference attacks [6, 33, 48, 60].

- *Message.* Message payloads contain a message for another node, intended to be passed up the protocol stack by the intended recipient, and encrypted with its public key.

- *Encapsulation.* Encapsulation payloads contain another payload, and are intended to be retransmitted after decryption and re-padding.

Upon receipt of a radiogram, every node adds the transmitter's public key (obtained from the radiogram) to a table of current network members. This table is used to select random nodes to route traffic through, as we will see soon. Next, every node attempts to decrypt the payload. Most of the time, the payload will not decrypt correctly because it is encrypted with some other node's public key or is a *cover* payload. If the payload does decrypt correctly and it is of type *message*, it passes it up the local protocol stack. If the decrypted payload is of type *encapsulation*, it re-pads it out to the message size of the received radiogram and adds it to its outgoing traffic queue.

Finally, whether the message decrypted correctly or not, the node consults its local randomized transmission control function to determine whether it is time to send the next radiogram in its output queue of which every node maintains one. A node will transmit whenever either of two conditions has occurred:

- If a timeout interval has passed since the last time it has received any traffic. The primary purpose of this timeout is to ensure that *cover* traffic is inserted in the network, to frustrate timing analysis or other inference attacks [33].

- After it has received a radiogram (whether it decrypted correctly or not) and a local random function determines that it is time to transmit. This random function will return "true" with probability approximately $1/N$, where N is a current estimate of the total number of nodes currently in range. When this occurs, the node waits a fixed interval (based on the bandwidth to be consumed by the network) before transmitting.

If a node has no outgoing traffic in its queue but is scheduled to transmit a radiogram anyway (*e.g.,* when its timeout has expired because its random transmission function returned "true") it sends a *cover* message. Otherwise, it pulls a message from its outgoing queue and sends it.

To initiate an outgoing message to some group member, a node selects at random a set of $M-1$ current group members, using the table of current network nodes (and their public keys) that was described earlier. It then creates an (undirected) "onion" of M encrypted encapsulated payloads, with the inner most payload encrypted with the destinations public key, and adds the whole package to its outgoing message queue.

Note that this protocol is very inefficient on modern computers because it requires a public key decryption operation on every received radiogram by every node. The value of this protocol is twofold: first, it is simple enough to prove basic properties about it (as we informally sketch below) and, second, it is useful as a building block for more efficient protocols.

Security Analysis. It is not hard to prove that (assuming employed encryption is secure), this protocol achieves source and destination anonymity to observer, for *any* adversary access structure $\mathcal{C}$ (not restricted in size). Complete anonymity (even when the destination is corrupted) is achieved as long as an adversary has not corrupted all M nodes in the onion. This is guaranteed if we choose $M > t$ (for shorter onions with $M \leq t$, the probability is $\binom{t}{M}/\binom{N}{M}$).

Anonymity per radiogram can also be proven, which implies that this protocol hides even the existence of communication. Moreover, even the sender of a radiogram in this protocol does not know who the receiver is, unless the sender is also the source who prepared the radiogram from scratch. This argument can be used to prove that an adaptive or pro-active adversary cannot gain much more information than a non-adaptive one in this protocol. Finally, as mentioned in Section 2, security against a Byzantine adversary can be achieved by adding authentication.

Note that (as discussed in Section 2), all these security guarantees are maintained for *any* scheduling of outgoing radiograms by the transmission control function, as long as it does not depend on the core traffic. Our choice of the randomized function above is designed to optimize round trip time.

3.2 Our Main Protocol

In this section, we describe our main protocol, which achieves the same anonymity guarantees as the protocol of Section 3.1.3, but with much better performance. The fundamental performance limitation of the above protocol is that it requires a public-key operation (decryption)

for each packet received, whether that packet was addressed to the node that received it or not. Ideally, such a public-key scheme would allow for efficient determination of whether the message would decrypt correctly, without requiring a node to perform a full decryption of the message. Unfortunately, we are aware of no public key cryptosystems that have this property (which we believe motivates an interesting open cryptographic problem). Our main protocol uses the protocol of Section 3.1.3 as a key distribution sub-protocol for a more efficient hybrid protocol. In effect, our protocol simulates a public key cryptosystem in which non-recipients need not do expensive trial decryptions for data they will be unable to successfully decrypt.

Essentially, we run two protocols in parallel: the protocol of Section 3.1.3 at low bandwidth (configured to occupy, *e.g.,* $\frac{1}{100}$ of the total channel computation or communication bandwidth), and a more computationally efficient version of the protocol that uses symmetric keys. Here, we refer to the part of the network running the low-bandwidth sub-protocol (from Section 3.1.3) as the *key distribution subnet* and that part running the efficient symmetric sub-protocol as the *traffic subnet*. As we show in the experimental evaluation in Section 5, the *traffic subnet* has a network throughput of at least an order of magnitude higher than that of the *key distribution subnet*.

When a new node joins the network, it must first join the key distribution subnet, using the same procedures and protocols as discussed in Section 3.1.3. Once part of the network, it identifies the other nodes in range and sends each of them via the key distribution protocol k unique (random) symmetric $(key, label)$ pairs (where the label is derived from a cryptographic hash of the key). It records each of these keys in a table, indexed by its label and the node to which it was sent. Observe that by using the public-key protocol, the receiver does not know which node originated the symmetric keys.

Upon receipt of a key distribution message containing $(key, label)$ pairs, the receiver records the keys and labels in a table, indexed by label. A node is free to delete both any keys it sent and any keys it received at any time from its tables, if it runs out of space; the number of keys stored is a configuration parameter. In general, extra keys should be deleted FIFO, but will be used (see below) LIFO.

The traffic subnet uses these keys to encrypt and route bulk message traffic. To send a message, a node selects a routing as in the key distribution protocol, and successively encrypts the message. This time, however, it uses the symmetric keys it sent to the nodes for the encryption. Each encrypted layer is prefixed with the label of the key used to encrypt it. Once a key is used, it is deleted from the table.

Upon receipt of such a message, a node checks the key label of the outermost encryption layer against its table of received keys. If the key is not in the table, the message is discarded. If it is, the message is decrypted and processed as in the protocol of Section 3.1.3. New symmetric keys are periodically sent via the key distribution subnet.

Observe that the processing of bulk traffic messages is very efficient here; a node can immediately determine whether it will be able to decrypt a received message, and only symmetric cryptographic operations are required even by the intended receiver. Security analysis of this protocol follows directly from that of the previous section; details are omitted here for brevity's sake.

Note the frequency and method of key distribution has some security implications. The strongest anonymity properties (equivalent to the basic protocol) are obtained if only one symmetric key is included in each message on the key distribution subnet and each such key is used to send at most one message on the traffic subnet. However, this carries with it a performance penalty, since that means that every radiogram on the traffic subnet has to have at least one corresponding radiogram on the key distribution subnet. Furthermore, certain anonymity-piercing attacks may be possible [32, 43, 51].

When more than one key is included in a given key distribution subnet message or when the same key is used to encrypt more than one message on the traffic subnet, it becomes possible for the receiver to link messages from the same sender together. This may be acceptable in practice, however, especially when multiple messages are part of the same logical flow or if new keys are generated frequently enough that only a limited number of messages are linked together. Allowing the same key to be used throughout a flow can greatly improve the overall efficiency of the network by permitting the key distribution subnet to be run at lower bandwidth.

4 Implementation

4.1 Simulation Environment

In order to get a feeling of how the various WAR protocols would behave, we created a simulation environment, implementing our constructions of Section 3 (with few optimizations). Our goal in constructing this prototype and simulation environment was to determine the behavior and computational impact/demands of our protocols on mobile devices.

Each node in the "radio network" is a PC connected to a common Ethernet LAN. Radiograms are simulated by multicast UDP packets; because of Ethernet limitations, radiograms can thus be at most 1472 bytes long (1500 bytes is the maximum Ethernet frame size, diminished by the 20 bytes of the IP header and the 8 bytes of the UDP header). Multicast packets carry the IP address of the sending node, and we can use this feature to infer the same sorts of information we could infer by doing direction-finding on a radio network: we know who the sending (real) node is, but we cannot know who the intended recipient is without actually decrypting the radiogram itself. In other words, an adversary sniffing the network would get similar information about the origin and destination of each packet as someone sniffing

a real-life wireless network with direction-finding equipment. Smaller radiogram sizes can be easily defined. As described in Section 3, each radiogram consists of a sequence of onion headers, followed by a payload, followed by random padding. Each onion header contains descriptive information, specifying whether this is an encapsulation header, a final message header, or a control header; a magic number to ensure that the header has been correctly decrypted when doing trial decryptions, the session key and IV under which the payload have been encrypted, and, if the header is a message header, the actual length of the useful payload.

4.2 Implementation Details

We are interested in evaluating end-to-end behavior of real network applications over an idealized WAR environment; for this reason, we wanted our implementation to give the illusion that access to the WAR environment looked like access to any other network infrastructure. In Unix terms, we needed a virtual interface with an IP address which could be used to send IP datagrams from and to. In order to avoid kernel-level development, which would be hard to debug, we used the FreeBSD *tun(4)* device driver; the driver links an entry in the */dev* directory to a network interface. Packets sent by any socket over, *e.g., tun0* can be read by a user-level process from */dev/tun0*; similarly, packets written to */dev/tun0* appear as if they had been received over the *tun0* interface.

More specifically, we assigned *10.10.0.0/16* as the subnet of our WAR network. On each node, the *tun0* interface takes an address from that subnet. For example, on node 13, we see:

```
$ ifconfig tun0
tun0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 560
      inet 10.10.0.13 --> 10.10.255.254 netmask 0xffff0000
      Opened by PID 58857
```

Note that, since the tunnel interface is defined as a point-to-point interface, a fake destination address is given. This is only used to trick the routing subsystem to route packets destined for the 10.10.0.0/16 subnet through the tunnel interface, as the following forwarding table extract shows:

```
$ netstat -r -n -f inet
10.10/16    10.10.255.254  UGSc  0  10  tun0
10.10.0.13  127.0.0.1      UH    0   0  lo0
```

There is a *war* process running on each node. The process reads from */dev/tun0*. Reads are atomic, in that the buffer given to the *read()* system call reading from the tunnel file descriptor must be big enough to hold one MTU's worth of data. Once the data are read, they are padded to a multiple of the (symmetric) encryption block size, a message radiogram header is prepended to the result, and a sequence of onion headers is also calculated. The onion is built as follows:

- For each header starting with the innermost (message) header:
  1) Generate a symmetric session key and IV.
  2) AES-encrypt using the generated key/IV pair.
  3) Place the key and IV in the header.
  4) Encrypt the header with the intended destination's public key.

Finally, the resulting packet is appended to the node's send queue, to be transmitted when its turn comes.

The *war* process also listens to UDP port 2003. The socket is set to ignore its own transmissions. When it receives a packet, it checks to see that it is of the right length (all radiograms must be of the same length). It then tries to decrypt the first header's worth of bytes with its own private key. If it fails, the radiogram was not meant for this node, and is discarded. If it decrypts correctly (as evidenced by the correct decryption of the magic number) the rest of the radiogram is decrypted using the session key and IV present in the header. If the radiogram was of type message, the length field in the radiogram header is checked, and that amount of bytes are sent to the */dev/tun0* device (so that they will appear as traffic received from the radio network to any process listening on the radio address). If it is of a type other than 'encapsulation', it is silently discarded. Otherwise, the header is discarded, the appropriate number of random bytes are appended to the end of the radiogram (so that the size is preserved), and the resulting packet is appended to the send queue.

4.3 Queue Disciplines

So far, we have described typical undirected onion routing. If packets are forwarded by each intermediate node as soon as they are received, some information about them could be gleaned by observing rapid trains of radiograms. To further protect against traffic analysis, we queue radiograms arriving or originating at each node, and transmit them at a constant rate. If there are none to be sent, the node will send cover traffic. To an observer, every node is transmitting a constant stream of radiograms, and no inferences can be made about the nature of the traffic. As we shall show in Section 5, this is a very heavy-weight protocol, and should thus only be used for key exchange.

First, let us observe that receiving a radiogram (even if not meant for the receiving node), is CPU-bound; it takes about $15ms$ on a 1GHz Pentium-class machine to decrypt with a 1024-bit RSA key, quite a difference from the submillisecond necessary to get a packet across the Ethernet, but comparable to the process of computing an onion of 3 to 6 layers (encryption being about an order of magnitude faster than decryption in our particular implementation). Note that, in principle, the use of hardware cryptographic accelerators can significantly improve the cryptographic operation latency and throughput [36]. We experimented with two queuing disciplines.

The first discipline does not do any queuing; as soon as a radiogram appears on the send queue, it is transmitted. As mentioned above, this discipline is undesirable, as we do not want the adversary to observe trains of radiograms appearing in quick succession and thus be able to infer the path of each onion (and thus each individual originator). However, the discipline provides a baseline of performance; this is the best we can do out of the available network and CPU resources. Since this queue discipline is only studied to provide a performance baseline, it does not provide any cover traffic.

The second queue discipline is a leaky-bucket: each node empties its queue at a constant rate, sending cover traffic if there is nothing in the queue. This queue discipline results in a uniform sharing of the available bandwidth by all the nodes; however, for this to work well, the number of nodes must be known in advance, and not change. Estimating the number of nodes is not hard; using feedback from the media access layer (a realistic assumption), we can know what fraction of the time the medium goes unused, or how many packet collisions occur; the sending rate can be adjusted to keep the medium full.

Notice that there is no need to randomize the order of messages in the queue. Since all nodes transmit at a constant rate, and cover messages are indistinguishable from legitimate ones, reordering would not add any security (and may cause problems at the network layer, as TCP does not react well to reordering).

What about the end-to-end behavior of protocols running on top of the radiogram network? The underlying network appears as a long, thin pipe. Applications that do not implement congestion-control mechanisms, such as ICMP or UDP, will lose packets when the network becomes overloaded. TCP, on the other hand, will interpret losses as resulting from congestion and back off, allowing the queues to drain. Because TCP works better when the round-trip delays are consistent, a queue discipline that encourages that should be preferred over one that allows round-trip times to vary wildly.

5 Performance Evaluation

While building a complete system requires evaluation in high-fidelity conditions (*i.e.,* using actual wireless devices), our goal here is more limited: we wish to determine the behavior and computational impact/demands of our protocols on mobile devices, without taking into consideration other aspects of the system that have no bearing on the protocols themselves.

We took performance measurements on a group of eight Pentium-III class machines running at 800MHz, on a 100Mbps shared (not switched) Ethernet. Although not faithful to a wireless environment, the shared Ethernet testbed allows us to simulate (imperfectly) a broadcast radio environment. We used a constant payload length of 560 bytes, large enough to contain 512 bytes of useful TCP payload, but also small enough so that small packets (such as TCP ACKs) would not be too wasteful of network resources. We evaluated each of the two queuing disciplines at three different RSA[6] key lengths (512, 768, and 1024 bits), with onions 0-, 1-, 2-, and 6-deep (0 meaning no onion routing was being used). The results are shown in Tables 1 and 2, and graphically in Figure 1. Note that Table 2 does not include results for 1024-bit RSA as testbed machines could not keep up with the required packet rate (25 packets/second).

Observe that the round trip times with such a low packet rate vary wildly. This is expected, since the machines are not running with synchronized clocks and even their own clocks have some jitter. The TCP throughput also has a very high variance, which is also expected; analyzing the effects of the particular queuing discipline on TCP however is beyond the scope of this paper. Note that both directions of a TCP connection were anonymized.

As can be seen from these results, the key distribution subnet is very slow. Especially when using cover traffic, at 25 packets/second per node, the TCP throughput drops significantly. The primary constraining factor in the key distribution subnet throughput is the computational overhead of RSA, as can be observed by the decline in throughput as we increase the key size. Using a public-key cryptosystem where the cost of encryption *vs.* the cost of decryption is more balanced would not have improved performance since all nodes try to decrypt all radiograms. Using higher-performance nodes would have improved throughput, at the expense of increased power consumption — a critical factor in MANET environments. Alternatively, at a more modest power consumption, we could make use of hardware cryptographic accelerators. While the typical accelerator does not significantly improve performance of public-key operations (due to the difficulty in parallelizing the underlying algorithm), as shown by Keromytis *et al.* [36], use of off-board crypto-processors would improve overall system performance by off-loading CPU-intensive operations to secondary logic.

The heavyweight protocol (key distribution subnet) is clearly unsuitable for sustained high-rate traffic. It should only be used to distribute session keys for use by the lightweight protocol (traffic subnet) that uses symmetric-key cryptography, described in Section 3.2. The performance of that protocol is given in Tables 3 and 4, and graphically in Figure 2. As can be seen by those experiments, the performance of the symmetric-key protocol is at least an order of magnitude higher than that of the public-key protocol, justifying our use of a two-tier protocol hierarchy. Even when using 6-hop onions with 250 packets/second *cover* traffic, the throughput achieved is sufficient to sustain a high-quality bi-directional voice communication channel (a typical application in military MANETs). Further performance improvements can be obtained through the use of hardware cryptographic ac-

[6] While RSA does not provide key anonymity [4], we used RSA for prototyping expediency.

Table 1: Key distribution subnet, no queuing

512-bit RSA			768-bit RSA		1024-bit RSA	
#L	RTT (ms)	TCP (Kbps)	RTT (ms)	TCP (Kbps)	RTT (ms)	TCP (Kbps)
0	10.6/10.7/10.9	456	21.3/21.9/26.3	188	46.2/48.4/53.5	83
1	19.9/20.5/21.1	220	41.0/42.7/46.5	84	91.0/93.3/101	39
2	30.3/31.5/35.7	119	61.6/66.7/76.0	63	133/141/146	26
6	68.7/73.5/79.4	58	146/149/154	23	315/326/332	11

Table 2: Key distribution subnet, with a queue discipline using 25 packets/second. In this experiment, our testbed machines could not encrypt packets fast enough when using 1024-bit RSA keys.

512-bit RSA			768-bit RSA	
#L	RTT (ms)	TCP (Kbps)	RTT (ms)	TCP (Kbps)
0	38/152/312	43	45/355/983	39
1	71/213/415	20	188/391/698	20
2	161/375/1075	12	243/602/930	15

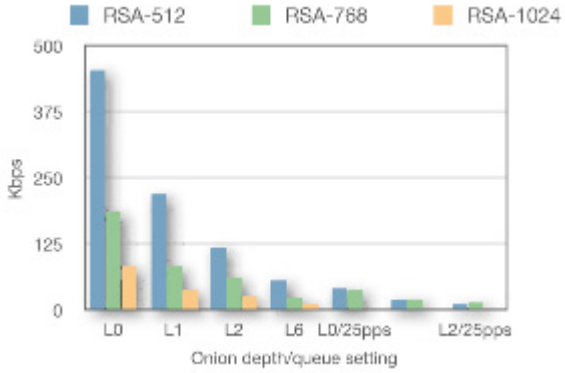

Figure 1: Key distribution subnet, both queue disciplines (summary of Tables 1 and 2).

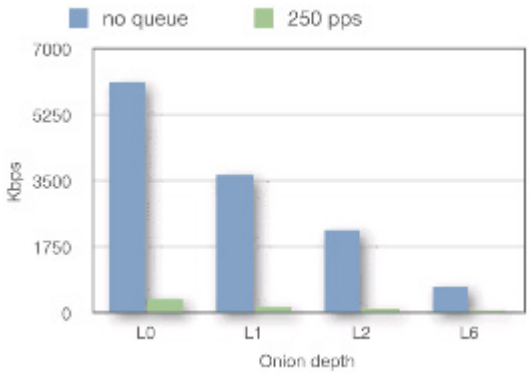

Figure 2: Traffic distribution subnet, both queue disciplines (summary of Tables 3 and 4).

celerators, since symmetric-key cryptography algorithms are highly amenable to parallelized processing [36].

6 Related Work

Previous work has identified a larger security problem in mobile ad hoc networks (MANETs) than with conventional wired and wireless networks [17, 69]. The main concerns are: (*a*) the use of wireless links makes certain types of attacks (eavesdropping, man-in-the-middle, denial of service, *etc.*) much easier in a MANET; (*b*) security solutions that take advantage of static configurations, such as the restrictions in the network topology that are exploited by traditional firewalls, are not applicable in a mobile environment; (*c*) MANETs depend on the cooperation of all nodes for their correct, or at least efficient, operation. Misbehaving nodes are typically difficult to detect and contain [46, 66]. Marti *et al.* apply intrusion detection to the problem of misbehaving routers in a MANET, while Deng *et al.* [17] take a preventive approach to the same problem. Finally, the nature of nodes that typically participate on a MANET (low computation and bandwidth capabilities, limited power budget) expose them to new attacks (*e.g.,* power exhaustion through repeated packet retransmission [3]) or increase their vulnerability to known attacks by making it difficult to adopt expensive mechanisms.

ANODR [37] is an anonymous on-demand routing protocol that hides network identifiers in multi-hop MANETs. The main challenge in such environments (in contrast to our network model of global broadcast) is enabling route discovery between two arbitrary nodes, while providing sender and/or receiver anonymity (and sender/receiver unlinkability). ANODR uses broadcast with trapdoor information, which is similar to our broadcast scheme, to transmit packets between nodes. The primary difference with our work is that in our scheme, the originating node selects (and varies) the transmission paths, whereas in ANODR the paths are dictated

Table 3: Traffic subnet using the symmetric-key protocol, no queuing.

#L	RTT (ms)	TCP (Kbps)
0	1.0/1.2/2.0	6120
1	1.6/1.9/2.7	3650
2	2.2/2.6/3.8	2200
6	5.3/5.8/7.5	700

Table 4: Traffic subnet using the symmetric-key protocol, with a queue discipline using 250 packets/second.

#L	RTT (ms)	TCP (Kbps)
0	7/28/58	360 (sdev: 72)
1	16/44/96	166 (sdev: 31)
2	48/142/522	119 (sdev: 28)
6	83/144/214	63 (sdev: 14)

by the wireless network topology. Thus, to achieve practical anonymity, MIXes (which introduce high latency) must be used. ANODR [37] requires a new public/private key pair for every forwarded message. Similar routing-based schemes are explored in other work [7, 54, 67, 68, 70]. Zhang *et al.* [67, 68] address both MAC- and network-layer anonymity in a multi-hop MANET by using pairing-based cryptography to generate a large number of pseudonyms per node. However Mask [68] contains, in plaintext, the final destination within every routing request message. Schemes similar to Mask are described by Lu *et al.* [42] and Lin *et al.* [41]. Boukerche *et al.* [8] propose using trust and reputation to avoid using untrustworthy nodes in their anonymous routing scheme.

None of these schemes uses *cover* traffic to protect against a global passive adversary over time. Yang *et al.* [65] extend ANODR by lowering the computation overhead (and eliminating the need for key exchange or any PKI-like infrastructure), at the cost of lower privacy guarantees (only source anonymity and routing privacy) that expose the message destination. In some scenarios that interest us (*e.g.,* battlefield), the message destination can be unacceptably revealing, especially when combined with packet-flow volume information (*e.g.,* a command HQ will be more "highly" connected). Seys and Preneel [52] propose establishing pairwise secret keys between neighboring MANET nodes, and use onion routes over these to route traffic anonymously. Their scheme is somewhat more complex than ours because it operates over a non-broadcast domain (*i.e.,* not all nodes can hear all transmissions). Chain-based Anonymous Routing (CAR) [53] uses a similar approach.

Jiang *et al.* [35] examine the problem of selecting routes from among various MIXes in a wireless *ad hoc* network. Jiang *et al.* [34] also study the use of per-flow *vs.* per-link cover traffic in a wireless *ad hoc* network, and concludes that the latter is less expensive in terms of required "dummy" packets, but requires encryption of each link. We extend their work for a broadcast medium with link-level encryption, which allows use of per-node constant-rate cover traffic, further reducing the amount of necessary dummy packets. Kong *et al.* [38] focus on protecting the mobility patterns of nodes in a geographical area.

Wu and Li [64] propose a model similar to ours, using onion routing in wireless mesh sensor networks to provide anonymity. However, because of the nature of the sensor network (all information is aggregated toward the gateway, and individual nodes are extremely lightweight), their scheme does not use cover traffic, making it susceptible to a passive global adversary attack. They propose using a ring structure [9] to hide the identity of the communication endpoints. Other related work uses multicasting along with incomparable public keys to provide receiver anonymity [61]. von Ahn *et al.* [58] propose the concept of "sender k-anonymity," whereby an attacker can only determine that the sender of a message was one among k entities.

Traditional anonymizing approaches focus on hiding the client's identity, and only take the *traffic rate* into consideration insofar as it facilitates traffic analysis by an observer. However, the fundamental assumption behind most of these approaches were that an adversary did not (or could not) have global view of the network. While this may be realistic in a wide-area wired infrastructure, it is inappropriate in a local-area MANET, where the attacker can eavesdrop and triangulate on any communication. Traditional approaches against traffic analysis attacks [27, 28, 30, 48, 56, 57] focus on individual links, based on the "wired infrastructure" assumption. The most basic anonymity solution is to interpose a proxy between two communicating parties [15]. The usefulness of this approach is limited to certain applications such as Web browsing under certain weak threat assumptions. However, various timing channels can be exploited to determine correlations between incoming and outbound traffic on such a proxy [6, 60].

Chaum's MIXes [13] were an early proposal to create an untraceable email system. The system was based on special-purpose nodes, called MIXes, which perform the anonymizing by re-ordering received messages and forwarding them through the MIX network. Under this approach, and subsequent work on various *remailers* that were put in service, an eavesdropper can only determine that a participant is communicating with the MIX. Traffic padding and message fragmentation are needed to protect against adversaries that can monitor the entire MIX network, as may be the case with a wireless network. DC-Nets [14, 59] is another proposal for constructing untraceable communication networks, based on oblivious coin flipping.

Mixes have been implemented for many types of communication, such as e-mail (*e.g.,* [29]), ISDN service [47], IP-layer infrastructure [44], and general synchronous com-

munication (including web browsing, see below). Mixes have also been used to anonymize location information in mobile telephony systems [21, 22]. Minx [16] is a cryptographic message format for encoding anonymous messages relayed through a network of Chaumian mixes, and is designed to protect against passive and active adversaries, as well as corrupt mix nodes. A more efficient version of MIXes that trades off complete mixing is discussed by Boneh and Golle [5].

ANON [39] is an IP-layer anonymizing infrastructure, allowing server addresses to be hidden from clients, and vice versa. It uses a set of network-resident nodes that act as anonymizing forwarders, similar to Chaum's MIXes. One basic assumption is that an attacker cannot monitor or subvert the whole ANON infrastructure itself, which is arguably realistic in certain scenarios (*e.g.,* in countering DoS attacks on servers) but inappropriate in a MANET. In follow-on work [40], they introduce the concept of *on-demand link padding*, which adds padding traffic based on the bandwidth usage observed from real traffic. In theory, this allows the amount of the padding to be limited. However, this approach will still reveal the source or destination of many traffic flows (*e.g.,* a military HQ), since it is only done on a per-link basis.

Onion Routing [25, 49] is an extension to MIXes that supports synchronous communication, such as web browsing. It uses nested encrypted addresses, called an onion, constructed by the initiator of a connection. The security of onion routing is analyzed by Syverson *et al.* [55]. Each successive onion router peels off a layer and forwards the connection. To avoid public key operations on a per-packet basis, onion routers use per-connection symmetric secret keys. Link padding is mentioned as a mechanism for countering traffic analysis. A connectionless approach that is otherwise similar to Onion Routing is Non-Disclosure Method (NDM) [20]. Tor, the second-generation onion router [18], also supports integrity protection, congestion control, and location-hidden services via rendezvous points.

Crowds [50] is a web-oriented peer-to-peer anonymizing infrastructure for synchronous communications. The main difference between Crowds and MIX-based solutions such as Onion Routing is that the routing path and the path length in Crowds are dynamic, versus static routing and preset path lengths in MIX networks. Analysis has shown that without active participation by the users of the anonymity system, there are attacks against anonymity that are more severe against Crowds than against MIXes [62, 63]. At the same time, there are attacks that work better against MIX-based solutions than against Crowds [23]. There are properties of both of these types of attacks that are specific to Web browsing on personal computers, so they are not directly relevant to our solution.

Tarzan [24] is an IP-layer anonymizing system that uses a peer-to-peer network to hide the client's identity and provides anonymity against casual eavesdroppers and small numbers of malicious participants. Tarzan also prevents global adversaries from determining which participant sent a particular message, but does not protect traffic sinks or hide the amount of traffic generated by a node.

7 Conclusions and Future Work

Anonymity has many potentially interesting applications in wireless networks, but conventional protocols do not work well in these environments. We have introduced a security model for wireless anonymity as well as a suite of protocols that provides basic anonymity functions in local broadcast networks. Our analytical and experimental results suggest that the protocols are realistic and sufficiently efficient to be useful in practice for many applications.

A number of interesting and significant problems remain, however. Admission control and network management is perhaps the most significant area here: how do we control network membership, especially in ad-hoc public networks, and how can we best link such networks together? How do anonymity networks perform under, and how can they adapt to, highly dynamic and difficult radio conditions (especially where there are many, mostly disjoint, users with only a few links between them)? And, of course, issues of scale are likely to be especially difficult. We believe the model and analysis we presented in this paper will serve as a useful launching pad to answering these interesting questions.

Acknowledgements

This material is based in part upon work supported by the National Science Foundation under grants CNS-07-14277, CNS-0627579, and CCF-05-41093. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris, "Link-level measurements from an 802.11b mesh networks," *Proceedings of ACM SIGCOMM*, pp. 121-131, Aug./Sep. 2004.

[2] A. Beimel, and S. Dolev, "Buses for anonymous message delivery," *Journal of Cryptology*, vol. 1, no. 16, pp. 25-39, 2003.

[3] J. Bellardo, and S. Savage, "802.11 denial-of-service attacks: Real vulnerabilities and practical solutions," *Proceedings of the* 12^{th} *USENIX Security Symposium*, pp. 15-28, Aug. 2003.

[4] M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval, "Key-privacy in public-key encryption," *Proceedings of the* 7^{th} *International*

Conference on the Theory and Application of Cryptology and Information Security, pp. 566-582, Dec. 2001.

[5] D. Boneh, and P. Golle, "Almost entirely correct mixing with applications to voting," *Proceedings of the* 9^{th} *ACM Conference on Computer and Communications Security (CCS)*, pp. 68-77, Nov. 2002.

[6] K. Borders, and A. Prakash, "Web tap: Detecting covert web traffic," *Proceedings of the* 11^{th} *ACM Conference on Computer and Communications Security (CCS)*, pp. 110-120, Oct. 2004.

[7] A. Boukerche, K. E. Khatib, L. Xu, and L. Korba, "A novel solution for achieving anonymity in wireless ad hoc networks," *Proceedings of the* 1^{st} *ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*, pp. 30-38, Oct. 2004.

[8] A. Boukerche, K. E. Khatib, L. Xu, and L. Korba, "Performance evaluation of an anonymity-providing protocol for wireless ad hoc networks," *Elsevier Performance Evaluation*, vol. 63, no. 11, pp. 1094-1109, Nov. 2006.

[9] M. Burnside, and A. D. Keromytis, "Low latency anonymity with mix rings," *Proceedings of the* 9^{th} *Information Security Conference (ISC)*, pp. 32-45, Aug./Sep. 2006.

[10] R. Canetti, and A. Herzberg, "Maintaining security in the presence of transient faults," *Proceedings of Crypto '94*, pp. 425-438, 1994.

[11] R. Canetti, and H. Krawczyk, "Analysis of key-exchange protocols and their use for building secure channels," *Proceedings of Eurocrypt '01*, pp. 453-474, 2001.

[12] R. Canetti, and H. Krawczyk, "Universally composable key exchange and secure channels," *Proceedings of Eurocrypt '02*, pp. 337-351, 2002.

[13] D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM (CACM)*, vol. 24, pp. 84-88, Feb. 1981.

[14] D. Chaum, "The dining cryptographers problem: Unconditional sender and recipient untraceability," *Journal of Cryptology*, vol. 1, no. 1, pp. 65-75, 1988.

[15] L. Cottrell, *The Anonymizer*. (http://www.anonymizer.com/)

[16] G. Danezis, and B. Laurie, "Minx: A simple and efficient anonymous packet format," *Proceedings of the ACM Workshop on Privacy in the Electronic Society (WPES)*, pp. 59-65, Oct. 2004.

[17] H. Deng, W. Li, and D. P. Agrawal, "Routing security in wireless ad hoc networks," *IEEE Communications*, vol. 40, no. 10, pp. 70-75, Oct. 2002.

[18] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," *Proceedings of the* 13^{th} *USENIX Security Symposium*, pp. 303-319, Aug. 2004.

[19] R. Draves, J. Padhye, and B. Zill, "Comparison of routing metrics for static multi-hop wireless networks," *Proceedings of ACM SIGCOMM*, pp. 133-144, Aug./Sep. 2004.

[20] A. Fasbender, D. Kesdogan, and O. Kubitz, "Variable and scalable security: Protection of location information in mobile IP," *Proceedings of the* 46^{th} *IEEE Vehicular Technology Society Conference*, Mar. 1996.

[21] H. Federrath, A. Jerichow, D. Kesdogan, and A. Pfitzmann, "Security in public mobile communication networks," *Proceedings of the IFIP TC 6 International Workshop on Personal Wireless Communications*, pp. 105-106, 1995.

[22] H. Federrath, A. Jerichow, and A. Pfitzmann, "MIXes in mobile communication systems: Location management with privacy," *Information Hiding*, pp. 121-135, 1996.

[23] E. Felten, and M. Schneider, "Timing attacks on web privacy," *Proceedings of the* 7^{th} *ACM Conference on Computer and Communications Security (CCS)*, Nov. 2000.

[24] M. J. Freedman, and R. Morris, "Tarzan: A peer-to-peer anonymizing network layer," *Proceedings of the* 9^{th} *ACM Conference on Computer and Communications Security (CCS)*, pp. 193-206, Nov. 2002.

[25] D. Goldschlag, M. Reed, and P. Syverson, "Onion routing for anonymous and private internet connections," *Communications of the ACM (CACM)*, vol. 42, no. 2, pp. 39-41, 1999.

[26] S. Goldwasser, and S. Micali, "Probabilistic encryption," *Journal of Computer System and Science*, vol. 28, no. 2, pp. 270-299, Apr. 1984.

[27] Y. Guan, X. Fu, D. Xuan, P. Shenoy, R. Bettati, and W. Zhao, "Efficient traffic camouflaging in mission-critical QoS-guaranteed networks," *IEEE Transactions on Systems, Man, and Cybernetics*, July 31, 2001.

[28] Y. Guan, C. Li, D. Xuan, R. Bettati, and W. Zhao, "Preventing traffic analysis for real-time communication networks," *Proceedings of the IEEE Military Communication Conference (MilCom)*, Nov. 1999.

[29] C. Gulcu, and G. Tsudik, "Mixing e-mail with BABEL," *Proceedings of the ISOC Symposium on Network and Distributed System Security (SNDSS)*, pp. 2-16, Feb. 1996.

[30] B. Hajek, and B. Radosavljevic, "Hiding traffic flow in communication networks," *Proceedings of the IEEE Military Communication Conference (MilCom)*, Oct. 1992.

[31] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung, "Proactive public key and signature systems," *Proceedings of the ACM Conference on Computers and Communication Security (CCS)*, 1997.

[32] A. Hintz, "Fingerprinting websites using traffic analysis," *Proceedings of the* 2^{nd} *International Workshop on Privacy Enhancing Technologies (PET)*, pp. 171-178, Apr. 2002.

[33] D. Huang, "Traffic analysis-based unlinkability measure for IEEE 802.11b-based communication systems," *Proceedings of the* 5^{th} *ACM Workshop on Wireless Security (WiSe)*, pp. 65-74, Sep. 2006.

[34] S. Jiang, N. H. Vaidya, and W. Zhao, "Routing in packet radio networks to prevent traffic analysis," *Proceedings of the IEEE Information Assurance and Security Workshop*, June 2000.

[35] S. Jiang, N. H. Vaidya, and W. Zhao, "Dynamic mix method in wireless ad hoc networks," *Proceedings of the IEEE Military Communication Conference (MilCom)*, Oct. 2001.

[36] A. D. Keromytis, J. L. Wright, and T. de Raadt, "The design of the openBSD cryptographic framework," *Proceedings of the USENIX Annual Technical Conference*, pp. 181-196, June 2003.

[37] J. Kong, and X. Hong, "ANODR: Anonymous on demand routing with untraceable routes for mobile ad-hoc networks," *Proceedings of ACM MobiHoc*, pp. 291-302, June 2003.

[38] J. Kong, D. Wu, X. Hong, and M. Gerla, "Mobile traffic sensor network versus motion-mix: Tracing and protecting mobile wireless nodes," *Proceedings of the* 3^{rd} *ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, pp. 97-106, July 2005.

[39] H. T. Kung, S. Bradner, and K. S. Tan, "An IP-layer anonymizing infrastructure," *Proceedings of the IEEE Military Communication Conference (MilCom)*, Oct. 2002.

[40] H. T. Kung, C.-M. Cheng, K.-S. Tan, and S. Bradner, "Design and analysis of an IP-layer anonymizing infrastructure," *Proceedings of the* 3^{rd} *DARPA Information Survivability Conference and Exposition (DISCEX)*, pp. 62-75, Apr. 2003.

[41] X. Lin, R. Lu, H. Zhu, P. Ho, X. Shen, and Z. Cao, "ASRPAKE: An anonymous secure routing protocol with authenticated key exchange for wireless ad hoc networks," *Proceedings of the IEEE International Conference on Communications (ICC)*, pp. 1247-1253, June 2007.

[42] R. Lu, Z. Cao, L. Wang, and C. Sun, "A secure anonymous routing protocol with authenticated key exchange for ad hoc networks," *Elsevier Computer Standards & Interfaces*, vol. 29, no. 5, pp. 521-527, July 2007.

[43] B. N.Levine, M. K. Reiter, C. Wang, and M. K. Wright, "Stopping timing attacks in low-latency mix-based systems," *Proceedings of the Financial Cryptography Conference (FC)*, Feb. 2004.

[44] NymIP, *The NymIP Effort.* (http://nymip.velvet.com/)

[45] R. Ostrovsky, and M. Yung, "How to withstand mobile virus attacks," *Proceedings of the* 10^{th} *ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 51-59, 2001.

[46] P. Albers et al., "Security in ad hoc networks: A general intrusion detection architecture enhancing trust based approaches," *Proceedings of the* 1^{st} *International Workshop on Wireless Information Systems*, Apr. 2002.

[47] A. Pfitzmann, B. Pfitzmann, and M. Waidner, "ISDN-Mixes: Untraceable communication with very small bandwidth overhead," *Proceedings of the GI/ITG Conference on Communication in Distributed Systems*, pp. 451-463, 1991.

[48] J. F. Raymond, "Traffic analysis: Protocols, attacks, design issues and open problems," *Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability*, pp. 10-29, 2001.

[49] M. Reed, P. Syverson, and D. Goldschlag, "Anonymous connections and onion routing," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 16, no. 4, pp. 482-494, May 1998.

[50] M. K. Reiter, and A. D. Rubin, "Crowds: Anonymity for web transactions," *ACM Transactions on Information System Security*, vol. 1, no. 1, Apr. 1998.

[51] A. Serjantov, and P. Sewell, "Passive attack analysis for connection-based anonymity systems," *Proceedings of the* 8^{th} *European Symposium on Research in Computer Security (ESORICS)*, pp. 116-131, Oct. 2003.

[52] S. Seys, and B. Preneel, "ARM: Anonymous routing protocol for mobile ad hoc networks," *Proceedings of the* 20^{th} *International Conference on Advanced Information Networking and Applications (AINA)*, Apr. 2006.

[53] R. Shokri, N. Yazdani, and A. Khonsari, "Chain-based anonymous routing for wireless ad hoc networks," *Proceedings of the* 4^{th} *IEEE Consumer Communications and Networking Conference (CCNC)*, pp. 297-302, Jan. 2007.

[54] R. Song, L. Korba, and G. Yee, "AnonDSR: Efficient anonymous dynamic source routing for mobile ad hoc networks," *Proceedings of the* 3^{rd} *ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, pp. 33-42, July 2005.

[55] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr, "Towards an analysis of onion routing security," *Proceedings of the International workshop on Designing Privacy Enhancing Technologies (PETS)*, pp. 96-114, 2001.

[56] B. R. Venkatraman, and R. E. Newman-Wolfe, "High level Pprevention of traffic analysis," *Proceedings of the* 7^{th} *Annual Computer Security and Applications Conference (ACSAC)*, Dec. 1991.

[57] B. R. Venkatraman, and R. E. Newman-Wolfe, "Transmission schedules to prevent traffic analysis," *Proceedings of the* 9^{th} *Annual Computer Security and Applications Conference (ACSAC)*, Dec. 1993.

[58] L. von Ahn, A. Bortz, and N. J. Hopper, "*k*-anonymous message transmission," *Proceedings of the* 10^{th} *ACM Conference on Computer and Communications Security (CCS)*, pp. 122-130, Oct. 2003.

[59] M. Waidner, "Unconditional sender and recipient untraceability in spite of active attacks," *Proceedings of Eurocrypt '89*, pp. 302-319, Apr. 1989.

[60] X. Wang, and D. S. Reeves, "Robust correlation of encrypted attack traffic through stepping stones by manipulation of interpacket delays," *Proceedings of the* 10^{th} *ACM Conference on Computer and Communications Security (CCS)*, pp. 20-29, Oct. 2003.

[61] B. R. Waters, E. W. Felten, and A. Sahai, "Receiver anonymity via incomparable public keys," *Proceedings of the* 10^{th} *ACM Conference on Computer and Communications Security*, pp. 112-121, Oct. 2003.

[62] M. Wright, M. Adler, B. Levine, and C. Shields, "An analysis of the degradation of anonymity protocols," *Proceedings of the ISOC Symposium on Network and Dsitributed System Security (SNDSS)*, Feb. 2002.

[63] M. Wright, M. Adler, B. Levine, and C. Shields, "Defending anonymous communications against passive logging attacks," *Proceedings of the IEEE Symposium on Security and Privacy*, May 2003.

[64] X. Wu, and N. Li, "Achieving privacy in mesh networks," *Proceedings of the* 4^{th} *ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, pp. 13-22, Oct. 2006.

[65] L. Yang, M. Jakobsson, and S. Wetzel, "Discount anonymous on demand routing for mobile ad hoc networks," *Proceedings of the* 2^{nd} *International Conference on Security and Privacy in Communication Networks (SecureComm)*, pp. 1-10, Aug./Sep. 2006.

[66] Y. Zhang, and W. Lee, "Intrusion detection in wireless ad-hoc networks," *Proceedings of the* 6^{th} *International Conference on Mobile Computing and Networking (MobiCom)*, pp. 275-283, Aug. 2000.

[67] Y. Zhang, W. Liu, and W. Luo, "Anonymous communications in mobile ad hoc networks," *Proceedings of INFOCOM*, Mar. 2006.

[68] Y. Zhang, W. Liu, W. Luo, and Y. Fang, "MASK: Anonymous on-demand routing in mobile ad hoc networks," *IEEE Transactions on Wireless Communications*, vol. 5, no. 9, Sep. 2006.

[69] L. Zhou, and Z. J. Haas, "Securing ad hoc networks," *IEEE Networks*, vol. 13, no. 6, Nov./Dec. 1999.

[70] B. Zhu, Z. Wan, M. Kankanhalli, F. Bao, and R. Deng, "Anonymous secur routing in mobile ad hoc networks," *Proceedings of the* 29^{th} *IEEE Conference on Local Computer Networks (LCN)*, pp. 102-108, Nov. 2004.

Matt Blaze is an associate professor of computer and information science at the University of Pennsylvania. His research focuses on cryptography and its applications, trust management, human scale security, secure systems design, and networking and distributed computing. He is particularly interested in security technology with bearing on public policy issues, including cryptography policy (key escrow), wiretapping and surveillance, and the security of electronic voting systems. He holds a PhD in computer science from Princeton University.

John Ioannidis ("JI") is the Chief Architect of Packet GENERAL Networks (http://www.packetgeneral.com/), a privately-held company providing regulatory-compliance products. In the past, he has been a researcher at Columbia University and AT&T Research. His interests revolve around ways to protect large complex infrastructures. A former Fulbright scholar, Ioannidis holds a PhD and MS from Columbia University and a diploma in Electrical Engineering from the University of Patras.

Angelos Keromytis is an Associate Professor with the Department of Computer Science at Columbia University, and director of the Network Security Laboratory. He received his B.Sc. in Computer Science from the University of Crete, Greece, and his M.Sc. and Ph.D. from the Computer and Information Science (CIS) Department, University of Pennsylvania. He is an associate editor of the ACM Transactions on Information and Systems Security (TISSEC). He recently co-authored a book on using graphics cards for security, and is a co-founder of StackSafe Inc. His current research interests revolve around systems and network security, and cryptography.

Tal Malkin is an assistant professor of Computer Science at Columbia University, where she directs the cryptography lab. She received her Ph.D. in Computer Science from the Massachusetts Institute of Technology in 2000, and joined Columbia after three years as a research scientist in the Secure Systems Research Department at AT&T Shannon Laboratory. Her research interests are in cryptography, security, complexity theory, and related areas. She has served on program committees and steering committees for over a dozen international conferences on cryptography, theoretical computer science, and security, she chaired the CT-RSA conference, and is on the editorial board for the Theory of Computing Journal. Prof. Malkin is the recipient an NSF Faculty Early Career Development award, an IBM faculty partnership award, a Columbia University Diversity Initiative Research Fellowship, and several research grants from NSF and other organizations.

Aviel D. Rubin is Professor of Computer Science and Technical Director of the Information Security Institute at Johns Hopkins University. Professor Rubin directs the NSF-funded ACCURATE center for correct, usable, reliable, auditable and transparent elections. Prior to joining Johns Hopkins, Rubin was a research scientist at AT&T Labs. He is also a co-founder of Independent Security Evaluators (securityevaluators.com), a security consulting firm. He is also the recipient of the 2004 Electronic Frontiers Foundation Pioneer Award. Rubin has a B.S, ('89), M.S.E ('91), and Ph.D. ('94) from the University of Michigan.